A pedagogical example: a family of stochastic cellular automata that plays Alesia

Nazim A. Fatès

A pedagogical example: a family of stochastic cellular automata that plays Alesia

Nazim Fatès
Université de Lorraine, CNRS, Inria, LORIA,
F-54000 Nancy, France
nazim.fates@inria.fr

August 2018

Abstract

Alesia is a two-player zero-sum game which is quite similar to the rock-paper-scissors game: the two players simultaneously move and do not know what the opponent plays at given round. The simultaneity of the moves implies that there is no deterministic good strategy in this game, otherwise one would anticipate the moves of the opponent and easily win the game. We explore how to build a family of one-dimensional stochastic cellular automata to play this game. The rules are built in an iterative way by progressively increasing the complexity of the transitions. We show the possibility to construct a family of rules with interesting results, including a good performance when confronted to the Nash-equilibrium strategy.

1 Introduction

The purpose of this note is to present a sketch on how stochastic cellular automata can be used to play a simple strategy game in which randomness has a central role. This game, named *Alesia* after the battle that opposed Gallic tribes and the army of J. Caesar 52 BC, has simple rules [**?**]: **(1)** The two players initially have the same number of soldiers, say 50. **(2)** Each round, the two players fight a *battle* by *simultaneously* engaging a given number of soldiers. The soldiers are then lost, whatever the outcome. Players must engage at least one soldier at each round (if they have not lost all their soldiers). **(3)** The winner of battle is simply the player who has engaged more troops; the front moves by one step in the direction of his opponent. **(4)** The winner of the game is the player who succeeds to reach his opponent's camp, that is, to make the front advance more than W steps, where W is fixed in advanced (here we take $W = 2$). **(5)** The game ends in a draw if none of the players reaches the opponent's camp.

To illustrate these rules, an example of game is given on Fig. 1-left.

sA	pA						
50	11	o	.	.	@	.	.
39	12	o	.	@	.	.	.
27	9	o	.	@	.	.	.
18	5	o	.	.	@	.	.
13	4	o	.	.	@	.	.
9	3	o	.	.	.	@	.
6	2	o	.	.	@	.	.
4	2	o	.	@	.	.	.
2	1	o	@	.	.	.	.
1	-	@	.	.	.	.	.

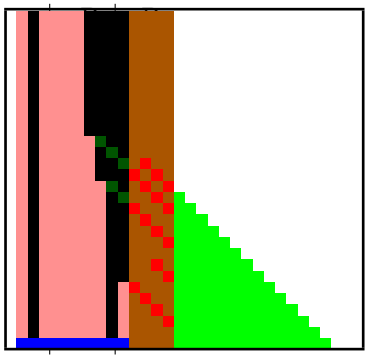

Figure 1: (left) Example of an Alesia game: sA, sB, pA, and pB are the soldiers and the plays of player A and B, respectively. (right) Example of space-time diagram produced by the `q2h2` player (see below). The cells in blue, brown, and green code the initial condition (states `A`, `F`, `B`) ; the result is given by the number of black cells (state `V`) ; other colours encode auxiliary states (see below for details). Time goes upward.

After playing a few games, one discovers that any good strategy should find a balance between playing too much or too little soldiers at each battle. Indeed, a large number of soldiers ensures victory but depletes the reserves for the next rounds. On the contrary, a small number of soldiers saves the reserves but increases the risks to loose the battle.

The other important point is that a good strategy is necessarily stochastic. Indeed, if for instance a player always plays 4 at the first round, the opponent can foresee this decision and decide to play 5 and thus win the first round. If the player always plays 20, the opponent's interest is to play only 1, and to deliberately loose the first round to gain an advantage in the number of soldiers. Of course, this argument can be recursively applied to the next rounds...

Our aim is to analyse whether simple cellular automata could calculate a good strategy for this game. Our motivation is to explore how "noisy" components with an elementary behaviour can cooperate to perform an interesting computation, as in a biological organisms. We are aware that cellular automata constitute a Turing-universal model of computation, and can thus compute any function that is computable by a classical machine, but our goal here is to make a decision emerge from *simple* non-deterministic mechanisms.

The links between cellular automata and game theory have been mainly explored with iterated two-player games on grids or graphs: these models distribute n players on a grid (or on a graph) and make the players interact by pairs with their neighbours. The players generally update their strategy according to the payoff received and the goal is to observe how the strategies dynamically evolve on the graph [**?**]. Interestingly enough, it can be observed that even small variations in the updating method (e.g., from synchronous to partially synchronous) may radically change the ultimate evolution of the system [**?**].

However, the use of cellular automata for the design of strategies is almost absent from the literature. Fraenkel has presented a pioneering work of how cellular automata may compute a particular strategy [**?**]. Cook, Larsson and Neary also made interesting connections between (deterministic) one-dimensional cellular automata and a generalisation of the game of Nim called the Blocking Wythoff Nim [**?**].

The purpose of this paper is to present some other simple examples by taking the Alesia game as a starting point. It can thus be considered as a *pedagogical* example in order to solve a problem in a distributed way. Our goal is not to reach optimality, but we will nevertheless compare our solutions to an optimal strategy in order to have a quantitative estimation of the quality of our cellular automata strategy.

2 Definitions

2.1 Formalisation of the game

We now introduce the formal definitions of the game. We assume that the game starts with N soldiers and that the "arena" is such that a player needs to make W (cumulative) steps in the direction of his opponent to win.

For the sake of simplicity we model a game by an infinite sequence of moves. Let a_t, b_t be the number of soldiers of player A and B, respectively, at time t and α_t, β_t the number of soldiers that they respectively engage at time t. We have:

$\forall t \in \mathbb{N}, a_{t+1} = a_t - \alpha_t$ and $\forall t \in \mathbb{N}, b_{t+1} = b_t - \beta_t$.

The position of the front evolves according to $w_0 = 0$ and:

$$w_{t+1} = w_t + \begin{cases} 1 & \text{if } \alpha_t > \beta_t \\ -1 & \text{if } \alpha_t < \beta_t \\ 0 & \text{otherwise.} \end{cases}$$

The game stops when one of the players hits the camp of his adversary or when there are no more soldiers. As the outcome of the game depends on the sequence of moves $s = (a_t, b_t)_{t\in\mathbb{N}}$, we call this sequence the *game* for the sake of simplicity.

Since each player is forced to play at least one soldier at each time step, the number of rounds is finite; we denote it by $T(s)$.

The gain $G(s)$ that results from a game s is 1, -1 or 0, if player A wins, or if player B wins, or if there is a draw, respectively. Formally, we have:

$$T(s) = \min_{t\in\mathbb{N}}\{w_t = W + 1 \text{ or } w_t = -W - 1 \text{ or } (a_t, b_t) = (0, 0)\},$$

$$\text{and : } G(s) = \begin{cases} 1 & \text{if } w_T = W + 1, \\ -1 & \text{if } w_T = -W - 1, \\ 0 & \text{otherwise.} \end{cases}$$

The rules of the game impose $a_0 = b_0 = N$ and: $\forall t < T$, $a_t > 0 \implies 1 \leq \alpha_t \leq a_t$, and $b_t > 0 \implies 1 \leq \beta_t \leq b_t$ and, for $t \geq T$, we set $\alpha_t = \beta_t = 0$.

2.2 Stochastic cellular automata

For the sake of simplicity, we will use one-dimensional cellular automata with nearest-neighbours interaction. To describe our model, we simply take an infinite line of cells : $\mathbb{Z}$. Note however that for the simulations, it is more convenient to use periodic boundary conditions ($\mathbb{Z}/n\mathbb{Z}$ with a large value for n).

We denote by Q the set of states the cells can hold. A configuration represents the global state of the system; it is an element of $Q^{\mathbb{Z}}$.

A stochastic cellular automaton is defined with a local transition function $\varphi : Q^3 \times Q \to [0, 1]$ and we write $\varphi\big((x, y, z), q\big) = p$ to express that a cell with a neighbourhood state (x, y, z) has a probability p to update to the state q. We require: $\forall (x, y, z) \in Q^3$, $\sum_{q \in Q} \varphi\big((x, y, z), q\big) = 1$.

The global transition function thus maps a configuration x^t to the configuration x^{t+1} such that:

$$\forall i \in \mathbb{Z},\ \Pr[x_i^{t+1} = q] = \varphi\big((x_{i-1}^t, x_i^t, x_{i+1}^t), q\big),$$

and where all the probabilities are drawn independently for each cell.

In order to build a strategy that uses a cellular automaton, we will translate the state of the game into a configuration, make this configuration evolve, and then interpret the resulting configuration as an action of the player. Given two players A and B, we will measure their *relative strength* with the expected gain that player A has against player B. This statistical estimation is obtained by repeating N games opposing A and B. If N_{A} and N_{B} are the number of games that A or B won, respectively, then the expected gain of A against B is given by $(N_{\mathrm{A}} - N_{\mathrm{B}})/N$.

We now present how we build a cellular automaton player by progressively improving the strategy players. This improvements are made by defining families of rules and making each new family compete against the previously found players. This method can be seen as a "bootstrapping" technique because each level of complexity emerges from a previously defined level of complexity.

3 Cellular automata players

All our players are Markovian: they base their decision on the current state of the game only. For the sake of readability, we will assume that the player we are describing is player A, while its opponent is player B. We recall our notations: a, b denote the number of soldiers of player A and B, respectively, and w denotes the position of the front at time t, the number of soldiers played by a given player Π is denoted by $\Pi(a, w, b)$. We also assume that the game is played in "standard" conditions, that is, with $W = 2$ and $N = 50$ [**?**].

Table 1: Expected gains of `uni-M` players opposed to other `uni-M` player. The deviation from an antisymmetrical form are due to the statistical variations (10^5 samples).

	`uni-10`	`uni-12`	`uni-14`	`uni-16`	`uni-18`	`uni-20`	`uni-22`	`uni-24`
`uni-10`		-0.133	-0.178	-0.182	-0.170	-0.146	-0.109	-0.057
`uni-12`	0.130		-0.068	-0.076	-0.060	-0.027	0.016	0.075
`uni-14`	0.177	0.069		-0.030	-0.017	0.016	0.061	0.122
`uni-16`	0.180	0.071	0.027		-0.006	0.018	0.068	0.133
`uni-18`	0.171	0.058	0.013	0.003		0.016	0.049	0.105
`uni-20`	0.148	0.026	-0.017	-0.023	-0.015		0.030	0.075
`uni-22`	0.108	-0.019	-0.065	-0.068	-0.053	-0.033		0.042
`uni-24`	0.060	-0.072	-0.124	-0.126	-0.103	-0.076	-0.038	

3.1 Uniform distribution players

The first player we can consider, denoted by `uniform`, plays a uniform number between 1 and a. It will serve as the basis of our bootstrapping process. Formally: $\texttt{uni-M}(a, w, b) = \mathcal{U}\{a\}$, where $\mathcal{U}\{0\} = 0$ and $\mathcal{U}\{k\}$ draws a random number uniformly in $\{1, \ldots, k\}$ for $k > 0$.

This strategy is easy to defeat with the following opponent: `beatUnif` always plays 1 unless the front is in position $-W$ (danger of loosing the game), in which case it plays b (as many soldiers as the opponent has). Formally: $\texttt{beatUnif}(a, w, b) = b$ if $w = -W$ and $\texttt{beatUnif}(a, w, b) = 1$ otherwise.

Our simulations show that `beatUnif` has an expected gain of 0.93 against `unif`: it wins the game with a probability greater than 95%.

3.2 Uniform distribution players with saturation

The weakness of `uniform` comes from the fact that it is too "generous": it is easy to "exhaust" simply by using a defensive strategy. A straightforward improvement of this player is to limit the maximum number of soldiers engaged at each time step. We define `uni-M` as the player which draws a number uniformly between 1 and a and then plays this value if it is lower than a threshold value M, or plays M otherwise. Formally: $\texttt{uni-M}(a, w, b) = \max\{\mathcal{U}\{a\}, M\}$.

The question then comes to know what is the best setting for M. Again, as there is no "absolute" good player; we thus simply oppose players with various settings of M and observe how they perform one against the other.

The results are presented in Tab. 1. The data represents the expected gain of `uni-M` players with different settings of M. These experiments indicate that a good setting of M is in the interval 16-18. Indeed `uni-16` and `uni-18` have a positive expected gain defeat when opposed to the other players and when opposed to each other, the difference in expected gain is not significant.

We are now in position to continue our bootstrapping process: our next objective is to build a CA player that performs better than `uni-18`. Before

Table 2: Expected gains of `binomial` players against `uniform` players (10^5 samples).

	`uni-8`	`uni-10`	`uni-12`	`uni-14`	`uni-16`	`uni-18`	`uni-20`
`bin-22`	0.554	0.409	0.370	0.375	0.368	0.377	0.391
`bin-24`	0.589	0.428	0.400	0.411	0.430	0.441	0.463
`bin-26`	0.608	0.433	0.399	0.417	0.451	0.475	0.501
`bin-28`	0.622	0.428	0.380	0.405	0.448	0.485	0.513
`bin-30`	0.628	0.420	0.345	0.368	0.420	0.468	0.504
`bin-32`	0.627	0.398	0.314	0.319	0.374	0.431	0.480

Table 3: Expected gains of `binomial` players against other `binomial` players (10^5 samples).

	`bin-15`	`bin-20`	`bin-25`	`bin-30`
`bin-15`		-0.182	-0.327	-0.417
`bin-20`	0.180		-0.002	-0.021
`bin-25`	0.328	0.003		0.098
`bin-30`	0.419	0.017	-0.099	
	`bin-20`	`bin-22`	`bin-24`	`bin-26`
`bin-20`		-0.004	-0.005	0.000
`bin-22`	0.006		0.015	0.039
`bin-24`	0.005	-0.018		0.035
`bin-26`	0.005	-0.030	-0.029	

going on, let us observe that "coding" the `uniform` or `uni-M` players with a one-dimensional cellular automaton is not that easy. Indeed, if the input is coded in the form of a configuration that has n cells in a given state, it is not clear how cells could interact *locally* in order to produce every possible output between 0 and $n-1$ (or n) with an *equal* probability.

3.3 Another simple player: the binomial player

If we have a set of cells which can hold a state with a given probability, the most intuitive "computation" is to draw a number according to a binomial distribution. We define the `binomial` players as the strategy where each soldier of the player has a probability ρ "to be played". For a given ρ, we set $R = 100*\rho$ and denote by `bin-R` the `binomial` with parameter $\rho = R/100$.

To encde our player, we simply use the binary alphabet $Q = \{\texttt{0}, \texttt{1}\}$ and map a game state (a, w, b), to the initial condition $x \in Q^{\mathbb{Z}}$ formed by a consecutive `1`'s on a background of `0`'s. Then each cell independently applies the rule where a `0` remains a `0` and a `1` remains a `1` with probability ρ and becomes a `0` with probability $1-\rho$. We have: $\texttt{binomial}(a, f, b) = card\{i \in \mathbb{Z}, y_i = \texttt{1}\}$, where y denotes the configuration obtained by a one-step transformation of x.

Table 2 shows the expected gain of various `binomial` players against various

`uniform-M` players. We observe that for each value of M, there is a different value of ρ which maximizes the expected gain. The player which has the *highest minimal* expected gain is `bin-24`. It is interesting to note that if we take $\rho = 0.28$, the "best" opponent of this `binomial` players is `uni-12` and *not* `uni-18`, as expected from what was seen by making `uniform-M` players together. This illustrates the fact that it is not possible to compare the expected gain of rules with a total order. Given three players A,B,C, we can observe that B performs better than C versus A, and nevertheless B is beaten by C.

When opposing the `binomial` players one against another, the best player is `bin-22` (Tab. 3). Our objective is now to find a player that beats this new challenger.

4 Taking into account the situation: the `q2h2` player

The previous model was not really a cellular automaton since there was no interaction between cells. The next improvement we can do is to take into account the position of the front and number of soldiers of the opponent.

We request that the initial state of the game to be translated in an initial condition of the cellular automaton with a simple method. Typically, the number of soldiers and the position of the front should be coded with a "unary" code, that is, each number of soldiers should correspond to the same number of cells in a given state, plus or minus some constants. Let us now present a player which respects these constraints, we name it `q2h2`.

The model employs 8 states: $Q = \{\texttt{E}, \texttt{A}, \texttt{V}, \texttt{V*}, \texttt{R}, \texttt{F}, \texttt{F*}, \texttt{B}\}$; its elements respectively represent the following states: empty, A-soldier, voluntary, voluntary-star, reluctant, front, front-star, B-soldier. The 'star' represents an information that travels from right to left in order to transmit an influence from the B-soldier cells to the different cells which represent the strength of player A. The following rules describe how this influence is transmitted.

We associate to a game position (a, w, b) an initial configuration x_{ini} built as follows : the a first cells are in state `A`. Next, we put δ cells in state `F`, where $\delta = W + w + 1$ represents the state of the front. The next following b cells are in state `B`. We thus have: $x_{\text{ini}}(a, w, b) = ..EE\underbrace{AAAAA}_{a \text{ times}}\underbrace{FFF}_{\delta \text{ times}}\underbrace{BBBBBBB}_{b \text{ times}}EEE...$

The evolution of the cellular automaton can be described with the following scenario: soldiers of player A (state `A`) turn to the voluntary state (`V`) or to the reluctant state (`R`) in one step. After that, some reluctant soldiers of A may turn to voluntary according to the "danger" they feel. This danger is evaluated as a combination of b and w: a) the more soldiers the opponent has, the greater the danger; b) the smaller the front is, the greater the danger. In practice, each cell in state `B` has a given probability (p_{T}) to initiate a signal that will travel to the left until it eventually reaches a cell in state `R` it then turns this cell to a `V`. This signal can also be absorbed with probability p_{A}.

As the information travels from the right to the left, we can define the local function $\varphi\big((x,y,z),q\big)$ with the use of the probabilistic function $\xi(q,q')$, which takes as an input the state of the cell itself q and the state of right neighbour q' and outputs a state in Q with a given probability. This function, which depends on three probabilities p_{V}, p_{T} and p_{A}, is defined as follows.

- An empty cell remains empty: $\xi(\mathtt{E},\cdot)=\mathtt{E}$.
- Each `A` immediately decides if it turns to a voluntary state or to a reluctant state: $\xi(\mathtt{A},\cdot)=\begin{cases}\mathtt{V} & \text{with probability } p_{\mathrm{V}},\\ \mathtt{R} & \text{with probability } 1-p_{\mathrm{V}}.\end{cases}$
- The behaviour of front cells and front-star cells depend on what they see on their right: a) a `B` cell: this corresponds to the case where the "stars" are initiated by the soldiers of the opponent. b) another `F` or `F*` cell : this case corresponds to the transmission of the star to the left. We set a probability p_{A} to be absorbed, i.e., the star is not transmitted. This reads:
  $\xi(\mathtt{F},\mathtt{B})=\xi(\mathtt{F*},\mathtt{B})=\begin{cases}\mathtt{F*} & \text{with probability } p_{\mathrm{T}},\\ \mathtt{F} & \text{with probability } 1-p_{\mathrm{T}},\end{cases}$ and:
  $\xi(\mathtt{F},\mathtt{F*})=\xi(\mathtt{F*},\mathtt{F*})=\begin{cases}\mathtt{F*} & \text{with probability } 1-p_{\mathrm{A}},\\ \mathtt{F} & \text{with probability } p_{\mathrm{A}}.\end{cases}$
  In all other cases, the front cells remains stable, the front-star cells become front cells: $\xi(\mathtt{F},q)=\mathtt{F}(\mathtt{F*},q)=\mathtt{F}$ for $q\notin\{\mathtt{B},\mathtt{F},\mathtt{F*}\}$. (Note that in a normal behaviour, the only useful case is $q=\mathtt{E}$.)
- A voluntary cell or voluntary-star cell simply transmits the star from right to left. This reads: $\xi(\mathtt{V},q')=\xi(\mathtt{V*},q')=\mathtt{V*}$ if $q'\in\{\mathtt{V*},\mathtt{F*}\}$ and $\xi(\mathtt{V},q')=\xi(\mathtt{V*},q')=\mathtt{V}$ otherwise.
- A refractory cell remains refractory unless it sees a star on its right. This is translated by: $\xi(\mathtt{R},\mathtt{V*})=\xi(\mathtt{R},\mathtt{F*})=\mathtt{V}$ and $\xi(\mathtt{R},q')=\mathtt{R}$ if $q'\notin\{\mathtt{V*},\mathtt{F*}\}$.
- Cells in state `B` simply disappear at the rate of one cell per time step. This reads: $\xi(\mathtt{B},\mathtt{E})=\mathtt{E}$ and $\xi(\mathtt{B},q')=\mathtt{B}$ if $q'\neq\mathtt{B}$.

An illustration of this behaviour can be seen on Fig. 1-right. (As it can be easily guessed, the front-star are drawn in red and the voluntary-star cells are in green.)

To analyse this rule, first, let us simply set $p_{\mathrm{A}}=0$, in other words, we do not take into account the state of the front. We ask how we can tune p_{V} and p_{T} in order to beat the `binomial` players. Recall that so far our best `binomial` player is `bin-22` ($\rho=0.22$). Table 4-left shows the result of this player against `bin-22` for different values of p_{V} and p_{T}. It can be seen that this player is easy to defeat: for example, for $p_{\mathrm{V}}=0.40$ and $p_{\mathrm{T}}=0$, one obtains an expected gain greater than 0. A zero value indicates that in fact, in this case it is sufficient to take into account only the strength of the opponent to obtain good results.

Table 4: Expected gains of `q2h2` players with $p_A = 0$ against `bin-22` (left) and `bin-35` (right). The two parameters p_T (columns) and p_V (lines) are varied (2.10^4 samples).

p_V \ p_T	0.10	0.20	0.30	0.40	0.50
0.0	-0.986	-0.189	0.470	0.637	-0.240
0.5	-0.565	0.133	0.530	0.346	-0.656
0.10	-0.070	0.264	0.507	-0.076	-0.864
0.15	0.021	0.342	0.343	-0.500	-0.953
0.20	0.062	0.328	0.023	-0.774	-0.989

	0.10	0.20	0.30	0.40	0.50	0.60
0.0	-0.997	-0.870	-0.212	-0.026	-0.246	-0.844
0.10	-0.415	0.007	-0.111	-0.311	-0.776	-0.984
0.20	0.107	-0.171	-0.382	-0.753	-0.971	-0.999
0.30	-0.205	-0.436	-0.735	-0.960	-0.998	-1.000
0.40	-0.501	-0.743	-0.948	-0.996	-1.000	-1.000
0.50	-0.778	-0.949	-0.997	-1.000	-1.000	-1.000

Table 5: Expected gains of `binomial` players (lines) against `q2h2` players (columns) with $p_V = 0.22$, $p_A = 0$ and a varying value of p_T (10^4 samples).

p_T	0.0	0.5	0.10	0.15	0.20	0.25	0.30
`bin-15`	-0.330	-0.541	-0.650	-0.709	-0.607	-0.276	0.277
`bin-20`	0.007	-0.114	-0.269	-0.389	-0.376	-0.102	0.323
`bin-25`	-0.008	0.086	0.058	-0.020	-0.048	0.083	0.387
`bin-30`	-0.082	0.015	0.157	0.206	0.203	0.286	0.478
`bin-35`	-0.192	-0.143	0.009	0.188	0.326	0.408	0.554

Note that the value of (p_V, p_T) which maximises the expected gain against `binomial` players varies greatly with ρ. For example, for `bin-35`, the best expected gain is obtained for $p_V = 0.20$ and $p_T = 0.10$, and this gain ($\sim$0.15) is much lower than for `bin-22`. This may seem paradoxical but remember that `bin-22` is the player which has the best result against the other `binomial` players, and that we know nothing for the other rules. All we can say is that we are sure that even with $p_A = 0$ the family of `q2h2` players *dominates* the family of `binomial` players: for every `binomial` player, there exists a `q2h2` player which can defeat it. This is a direct consequence of the fact that the `q2h2` players include all the `binomial` players simply by setting $p_T = 0$.

Table 5 show how various `binomial` players can be defeated by fixing $p_V = 0.22$ and setting an appropriate value of p_T. Similar results can be obtained for other values of p_V, which confirms the great advantage of the `q2h2` players against the `binomial` players.

Despite these encouraging results, we could not find any setting of (p_V, p_T)

Table 6: Expected gains of `q2h2` players with $p_T = 0.25$ against `bin-30` (left) and against `Nash` (right). The two parameters p_A (lines) and p_V (columns) are varied (10^5 samples).

	0.10	0.15	0.20	0.25
0.20	0.111	-0.065	-0.256	-0.370
0.40	0.254	0.208	-0.010	-0.213
0.60	0.280	0.348	0.178	-0.027
0.80	0.276	0.395	0.272	0.081
0.100	0.267	0.403	0.280	0.108

	0.0	0.2	0.4	0.6	0.8	0.10
0.20	-0.050	-0.058	-0.075	-0.095	-0.115	-0.148
0.40	-0.039	-0.033	-0.043	-0.056	-0.063	-0.088
0.60	-0.035	-0.029	-0.027	-0.029	-0.039	-0.048
0.80	-0.093	-0.051	-0.038	-0.029	-0.027	-0.038
0.100	-0.183	-0.139	-0.081	-0.046	-0.043	-0.033

which would dominate all the other `binomial` players. Once again, the difficulty stems from the impossibility to establish a total order between rules. For example the rule with the setting $(p_V, p_T) = (0.5, 0.8)$ defeats `bin-22` (with an expected gain of 0.60) and is defeated by `bin-35` (with an expected gain of 0.67)... but, as seen above, `bin-22` defeats `bin-35`!

In a second step, to demonstrate how setting a positive value of p_A can be useful, simply examine a precise situation and leave a more systematic study for future work. In the paragraph above we showed how setting $p_A = 0$ and varying p_V and p_T allows us to defeat every `binomial` player. However, against `bin-30`, only a small region of (p_V, p_T) has a positive expected gain, and the maximum positive gain one can obtain is around 0.1.

Table 6-left shows the expected gain of `q2h2` against `bin-30` when we set $p_T = 0.25$ and vary p_A and p_V. It can be seen that allowing $p_A > 0$ gives much better results: in particular for $p_V = 0.15$ and $p_A = 0.8$, the expected again is above 0.4, which is quite impressive.

We have seen that it is not possible to totally order the players according to their respective scores. There exists however a strategy which is never defeated (on average). We denote this player by `Nash`, as it corresponds to what is called the Nash equilibrium: `Nash` gives a guarantee to have a positive expected gain against any player `P`, but against a third player `P'`, it may be that `P` performs better than `Nash`.

We have tested the performance of `q2h2` against `Nash`, with a setting of $p_T = 0.25$ and different values of p_V and p_A. The results are displayed on Tab. 6-right: we see that in some cases, the average expected loss is close to zero and of the same amplitude as the noise on the measures. In practice, this implies that one can hardly distinguish between the `Nash` player and the `q2h2` player with the proper settings.

These first results are rather encouraging and there are many directions in which they can be deepened. For example, it is interesting to examine the scaling properties of our cellular automata: obtaining the optimal Nash-equilibrium player demands more time as the size of the game increases, our models can easily be applied to larger sizes, maybe with an adjustment of the three probabilities. Of course, the behaviour of this cellular automaton can also be obtained

with classical mathematical functions but here we wanted to examine how simple interacting elements would play this game. Our goal was to show that a non-trivial behaviour can be obtained by progressively increasing the complexity of the rules. Another research direction would be to make the system evolve autonomously and see if it can discover new levels of complexity without an external aid.

acknowledgements

We express our sincere gratitude to Bruno Scherrer for introducing us to the game of Alesia and for providing us with an optimal Nash player program. We thank Irne Marcovici for her valuable comments on the manuscript.